A Travel Consultation System : Towards a Smooth Conversation in Japanese

H. Suzuki, M. Kiyono, S. Kougo, M. Takahashi, S. Motoike, and T. Niki

Tokyo Systems Research Department
Matsushita Electric Industrial Co., Ltd.
17-15, Shinbashi 6-chome, Minato-ku,
Tokyo 105, JAPAN

ABSTRACT

We have developed a travel consultation system which extracts user's requests through natural language (Japanese) conversation and answers destinations that fit the case.

The system has four parts, the parsing subsystem, the user model subsystem, the sentence generator, and the database retriever. All subsystems are written in PROLOG.

The distinctive characters of conversations which our system makes are:
- smoothness and kindness,
- that the system can be a topic introducer, and
- efficiency.

How the system works is as follows:
First, the system parses Japanese sentences that the user inputs. The input sentences are interpreted as user's requests, not as the answer for the previous question. So he can add any requests other than the questioned one. Next, the system guesses the user's type, evaluating user models that the system has. Each user model represents the expected type of users. Then it generates a next question which that type of users would care. This parse-guess-generate sequence are continued till the topics which the user cares are exhausted. At last the system searches the database and shows places that satisfy all requests of the user.

OUR OBJECTIVES AND A SAMPLE DIALOG

Our research objective is to make a 'user friendly' interface. But what 'user friendly' means is a difficult question. Our approach is to create a machine that can communicate by the means that man uses everyday, *i.e.* a natural language conversation.

Most distinguishing characteristics of human conversation by natural language are its *efficiency* and *smoothness*. Grice(1975) argues that there are underlying constraints, and proposes two kind of rules, the cooperative principle and four maxims of conversation. Neustupný(1982) also argues constraints under communication, and proposes nine rules.

S：御希望をいれてください。	Enter your demand please.
U：７日間の新婚旅行に行きたいのです。	We'd like to go 7-day honeymoon.
予算は２人で５０万円です。	We estimate the cost about 500,000yen.
新婚さんの多い所がいいです。	We'd like to visit where many honeymooners go.
S：どのような所に行きたいですか。	What kind of place do you want to visit?
U：ロマンティックな涼しい都会に行きたい。	We want to visit a cool and romantic city.
S：泊まるところは高級ホテルがいいですね。	You plan to stay at a high class hotel, don't you?
U：はい。	Yes.
S：出発はいつごろですか。	When do you want to depart?
U：６月に東京から行くのです。	We'd like to depart from tokyo in June.
S：交通手段は何がいいですか。	By what do you want to go?
U：え。	Eh?
S：飛行機とバスと車と電車と船から選べます。	Choose one from an airplane, a bus, a train, and a ship.
U：飛行機。	An airplane.
S：遠いところがいいですね。	You want to go far, don't you?
U：そうです。	Yes.
S：わかりました。しばらくおまち下さい。	All right. Wait a minute.
S：札幌をお勧めします。	We recommend SAPPORO.

Figure 1. An example of dialogs our system made

For to make computer programs keep these rules and constraints, we have to study about six points:

(1) Selection of styles
(2) Generation without ambiguity
(3) Cooperative answering
(4) Reasoning user's answers
(5) Selection of topics
(6) Clarification subdialog

We have implemented some restricted aspects of above six features in our system. Figure 1 shows that our system has the feature (2), (4), (5) and (6). For example, our system does not ask "Where do you want to stay?", but asks "You plan to stay at a high class hotel, don't you?". This is an example of feature (4). Another is an example of clarification subdialog, feature (6). When the user says "Eh?", the system finds out that the user cannot understand the question and clarifies its idea by listing all possible answers.

But what we would like to mention in this article is our new proposal about how to choose next topic at the conversation. So in the next section, our user models are introduced, and

the simplified algorithm for selecting topics which employs user models are introduced in the following section.

USER MODELS CONVEYED TO SELECT TOPICS

We employed user models to select next topics in conversations. We summarize our idea and algorithm here. Detailed descriptions are found in (Suzuki 1985).

We propose to construct user models based on attitudes towards topics. We respected the following five attitudes.

(1) Whether the user cares about such topics. Or he has something to say about a topic or not.
(2) Whether he thinks he needs to say about such a topic or not.
(3) Whether he minds talking about such topics or not.
(4) Whether he can introduce such topics or not.
(5) Whether he remembers a topic or forgets it.

Using these, we classify topics into fourteen groups, and get five classes. Five classes are U, S, SD, R, and X. Class U consists of topics that a user has something to say about, that he thinks he need to say, that he doesn't mind talking about, that he can introduce into the conversation, and that he doesn't forget about. So topics in class U are introduced at the earliest chance by the user. Topics in class S and SD lacks some feature mentioned above, but the user doesn't mind talking about it, anyway. So these topics are to be introduced by consultant, the system. The user expects that the system knows the answer for topics in class SD, *i.e.* he thinks he need not talk about such topics. Topics in class R are topics which the user has nothing to talk about it. Topics in class X are forbidden topics that the user does mind talking about it for various reasons.

So there are two sorts of topics, topics that are to be talked about in the conversation and topics that are not to be mentioned. Topics in U and S are in the former sort, and topics in R and X are in the latter sort. Topics in SD can be considered in two ways. But in this article they are treated in the former sort, *i.e.* they are asked in the tag question form to get the user's confirmation.

A basic user model consists of these topics in the former sort. A user model is a basic user model with restrictions of possible answers for topics in the model.

SYSTEM CONFIGURATION

Figure 2 shows our system's configuration. Our system consists of four subsystems. We

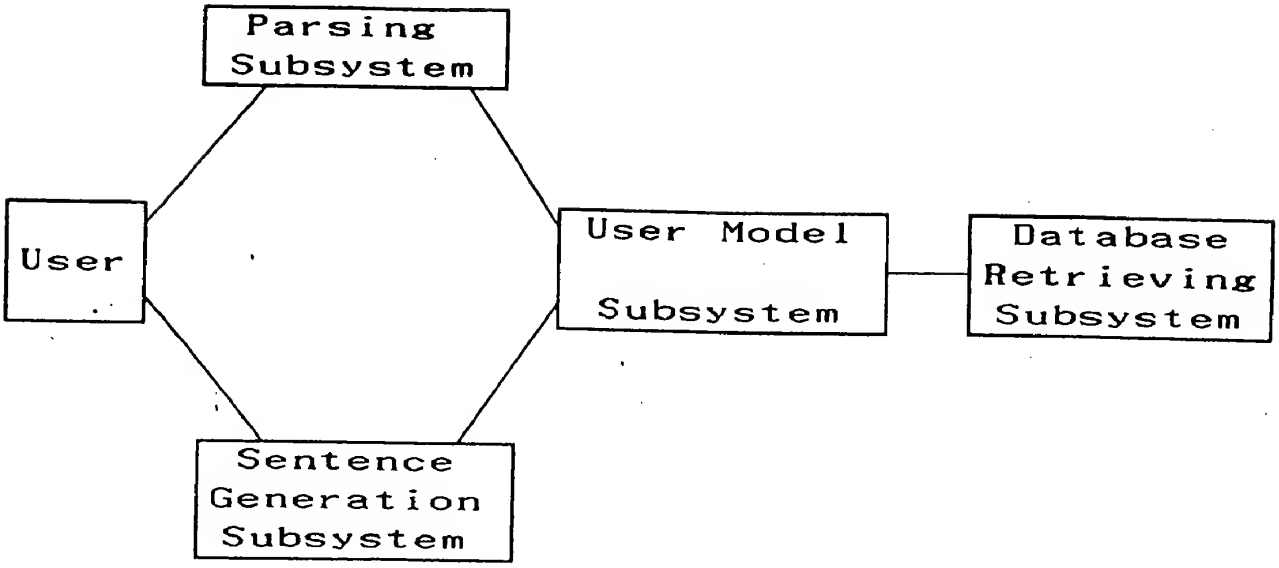

Figure 2. System Configuration

will explain each subsystem in this section.

Parsing Subsystem of Japanese

This parser/analyzer of Japanese sentence has three main job steps.
Step1: Analyzes a given sentence and generates its structural description.
Step2: Based on the structural description above, outputs the meaning representation of the sentence.
Step3: Extracts the attribute-value pairs from the meaning representation.

At step 1, we employed the LFG parser in PROLOG (Yasukawa 1983). But we developed the original grammar. The basic idea of our grammar owes much to Teramura(1982). A rough sketch of our grammar is as follows:

- A sentence has two part, the sentence kernel and the auxiliary part. In the sentence kernel, objective facts are described, and in the auxiliary part, speaker's attitudes towards that facts and tenses and aspects are represented.
- The structure of a sentence kernel is X^*V, where X corresponds to a chunk such like a noun phrase with a case particle or adverbial phrase (we call it "hogo") and V corresponds to a chunk such like a verb phrase (we call it "jutsubu").
- A auxiliary part consists of particles that represent tenses, aspects, and speaker's attitudes. The order of them is strictly determined.

Our grammar mainly treats the following four linguistic features of Japanese.
(1) Case government by verbs
(2) Modality
(3) Tense/Aspect particle
(4) Voice and transformation of case government

The analyzing strategy of these features uses the classification of Japanese verbs. To treat the first feature, we described necessary cases in the lexicon of verbs. Here, cases mean surface cases, not deep cases nor gramatical functions. One of the reason why we do not simply apply the LFG theory to Japanese is that in Japanese, any constituent of a sentence can appear at any place before verbs, and even more they can be omitted.

One of the problems about the third feature is that a tense/aspect particle '*teiru*' has two different meaning, progressive tense and stative tense. These problems are treated by the agreement of grammatical feature of verb and adverbs. Voices are solved at lexical level except causative. In Japanese, there are causative particles 'seru' and 'saseru'. But in our grammar, they have their own case government like verbs and make a F-structure.

At step 2, we constructed a meaning representation according to F-structure and its 'sem' feature. Our meaning representation is somewhat situation semantics like, but generating algorithm differs very much from their Aliass described in the book (Barwise 1983). Settings are not used, either. The detailed algorithm is described in our earlier paper (Suzuki 1983).

At step 3, we prepared a list of event-types and roles for each attribute used. If the situation is of that type, then the pair of the attribute and the value that is anchored to the role is generated. Then the subsystem outputs a collection of pairs.

Sentence Generation Subsystem

The sentence generator of Japanese also has three job steps.
Step1: Given an attribute and values, if any, generates its deep case structure description.
Step2: Based on the structure description above, outputs the surface case structure in a grammatical word order.
Step3: Makes morphological adjustment such as verb inflections.

At step 1 and 3, most works are done by lexicons, dictionaries, and tables. No new methods are employed at these steps.

Step 2 is the most significant core of sentence generator. Details are described in our paper (Takahashi 1985). The points of our method are three new data/programs listed below.

(1) A Japanese grammar in a form of transformation rules between subtrees.
(2) A translator which pre-compiles each transformation rule into a Horn clause, with the interpretation of such transformation rules.
(3) An engine which divides a tree into subtrees and reconstructs a tree from the transformed subtrees.

There are basically three kinds of transformation rules in our grammar, addition rules, deletion rules, and movement rules. (Inoue 1976; Shibatani 1978) Our grammar has about 150 rules of such kinds.(Mizutani 1983) In rules, capitalized identifiers, except those starting with the letter 'C', stand for part variables which may unify any structure including a sequence of subtrees. We use '@' and '&' for describing feature checking, and '*' for designating main category of the rule. if '*' is omitted, the top node is considered as the main category. No rules are applied twice to the same main category to avoid infinite loop.

The translator pre-compiles each transformation rule into a Horn clause. The Horn clause, as a PROLOG predicate, means a relation between two trees, an input tree and an output tree. So its body has two parts, one for checking whether an input tree matches the left-side pattern of the rule, and the other for rebuilding the output tree by using the constituents got from the above analysis. A collection of these predicates forms a transformation part of this subsystem. Therefore, at execution, transformations are directly done by PROLOG interpreter/compiler. In this sense, our method is efficient.

The engine part is written in PROLOG. It reduces the input tree to the minimum subtrees and sends them sequentially to the transformation part till no transformation rules are applicable. Then it reconstructs the tree by using the new subtrees.

Figure 3 shows an example of our grammar rules and the result of their translations. This example is an addition rule describing : "if a verb has the feature 'motive', then add the case particle '*ni*' after the noun phrase whose case is locative."

User Model Subsystem

This subsystem has two main jobs, selection of next topics and default reasoning for answers. These are realized by a concurrent algorithm which employs three kind of processes:

- The General Model Process
- User Model Processes
- The Decision Process

The general model process has:

- The set of all topics

```
joshi : [CX1, X11,
            [pred, X21, [doushi@[motive], D], X22],
            X12,
            [case, X23, [*place, X31], X24],
            X13] ==>
        [CX1, X11,
            [pred, X21, [doushi, D], X22],
            X12,
            [case, X23, [place, X31, [joshi, ni]], X24],
            X13].

rule(A,B,C,A,B,D,joshi,E,[F|[joshi([ ],[ ],ni)]]) :-
        divide(C,pred(G,H),I,J,K),
        divide(J,doushi(L,M),N,O,P),
        intersect([motive],M),
        divide(K,case(Q,R),S,T,U),
        divide(T,place(V,W),X,Y,Z),
        not_member(joshi,V),
        !,
        divide(D,pred(G,H),I,A1,B1),
        divide(A1,doushi(L,M),N,O,P),
        divide(B1,case(Q,R),S,C1,U),
        divide(C1,place([joshi|V],W),X,D1,Z),
        divide(D1,joshi([ ],[ ]),Y,F,[ ]).
```

Figure 3. An example of transformation rules and their translations

- Default values with applicable conditions

A user model process has data of three kinds:

- A set of topics
- Constraints on values
- Default values with applicable conditions

Figure 4 shows an example of a user model. Our system reads these descriptions and makes PROLOG programs for user model processes automatically.

The decision process are implemented as follows:
Prepare three classes, 'white', 'grey' and 'black'. Put all user models in the white class. If the input are not expected, *i.e.* not in the set of attributes, the model's class goes down one rank. If a model of black class goes down, it vanishes away. The order in a class is determined mainly by evaluating correctness of the default. The top model of the highest class is the current user model, and a topic which is nearest to the current topic is expected as the next topic.

```
title       honeymoon ::
attribute meimoku,
          mood,
          budget,
          ......
          goby ::
constraint meimoku : [honeymoon],
          ...... ::
expect    mood : [romantic],
          budget > 10 ↦ goby : [airplain] .
```

Figure 4. An example of a user model description

As a total, this subsystem works as follows. First, attribute-value pairs are received by the decision process. Then it evaluates the rank of each user model. While evaluating, the candidates for the next topic are also calculated by each user model. Then the decision process adopts the candidate of the highest rank, or the general process if no user models survive.

Database Retrieving Subsystem

The database retrieving subsystem has two parts, the database about destinations and the retrieving system with routing. Entities of the database describe the features of places declaratively like

data(temperature, warm, okinawa).

So almost all process of retrieving are done by unification facility of PROLOG . But there remains some to be considered. For example, cost of the travel depends not only on the destination but also on starting place and routing, so our retriever has a simple routing process and cost estimating process. Another example is the problem about seasons. One can play golf at Sapporo in summer, but cannot do it in other three seasons. The solution is to translate these constraints into Horn clauses and to check them when retrieving.

CONCLUSIONS AND FUTURE WORKS

Comparison with other systems

Our main concern and interest are to realize a *smooth* conversation. And our system can make a conversation with two features,

(1) Both the system and the user can take the initiative to select new topics.
(2) The system drives the conversation to the end efficiently.

SCHOLAR (Carbonell 1970) also embodies the first feature, that both can take the initiative in conversation. But as for SCHOLAR , the situation is rather different. SCHOLAR is a CAI system of geometry, so usually the system takes the initiative. Only when a user cannot answer questions that SCHOLAR asks, he may take the initiative and can ask a question. The system takes the initiative usually, and a user takes the initiative only in sub-dialogs. So a user cannot take the initiative of the main conversation. But our system has no such limitation. A user can take the initiative at any time during the conversation, by introducing new topics in his answering phase. Then the flow of the topic will change and go for that direction.

GUS (Bobrow 1977) can make a cooperation with the user to drive the conversation to get at the feasible answer. But what GUS does is to use the frame that represents the logical construction of the problem to guide the user. Its course of conversations are fixed and rigid. Our user model represents not only logical constructions but also user's types. Using models, our system can perform more flexible conversation than GUS can perform. Even more, our system's conversation has a property of 'efficiency'. Using models enables the system to anticipate user's answers. We can make yes-no questions rather than just ask him about the topic.

Future Works

We must respect many, at least three, more features about selecting topics.
(1) We considered nothing about heuristic knowledges and rules in the QA field when selecting topics.
(2) If the consultant is human, sometimes, especially at the final stage, he selects the topic which narrows the range of possible answers effectively.
(3) Our algorithm always in the model-selection mode. Humans, however, fix the model so that the conversation become robust to mistakes like slip of the tongue.

The implement method of these features are currently studied.

Other subsystems also have many features not implemented. As for parsing subsystem, analyzing anaphors, including zero anaphors, are very big open problem we have to attack. To generate a natural sentence, mechanisms to respect discourse grammar like word ordering according to the context are needed. These are now in study.

ACKNOWLEDGMENTS

The Authers are grateful to the members of natural language processing group of ICOT Research Center and the members of AIUEO for their valuable advice. We also thanks to Dr. Noda, Mr. Suzuki, and Mr. Komorida, for their continuous encouragement and various comments.

REFERENCES

Barwise J, Perry J (1983) Situations and Attitudes. MIT Press, Cambridge, Mass.

Bobrow DG, Kaplan RM, Kay M, Norman DA, Thompson H, Winograd T (1977) GUS, A Frame-Driven Dialog System. Artificial Intelligence 8: 155-173

Carbonell JR (1970) AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction. IEEE transaction on Man-Machine Systems MMS-11: 190-202

Coulthard M (1977) An Introduction to Discourse Analysis. Logman, London Grice HP (1975) Logic and Conversation. In: Morgan JL (ed) Syntax and Semantics III: Speech Acts. Academic Press, New York, p 41

Inoue K (1976) Henkei Bunpou to Nihongo ('Transformational Grammar and Japanese'). Taishukan, Tokyo (in Japanese)

Kaplan RM, Bresnan J (1982) Lexical-Functional Grammar. In: Bresnan J (ed) Mental Representation of Grammatical Relations. MIT Press, Cambridge, Mass., p 173

Mizutani S, Ishiwata T, Ogino T, Kaku N, Kusanagi Y (1983) Bunpou to Imi I ('Grammar and Semantics I'). Asakura, Tokyo (in Japanese)

Neustupný JV (1982) Gaikokujin-tono Communication ('Communication with Foreigners'). Iwanami, Tokyo (in Japanese)

Shibatani M (1978) Nihongo-no Bunseki ('Analysis of Japanese'). Taishukan, Tokyo (in Japanese)

Suzuki H (1984) Nihongobun-no Imi-no Joukyouimirontekina Kijutsu ('Event Types Represent Meanings of Japanese Sentences'). Information Processing Society of Japan, WGNL 42-3 (in Japanese)

Suzuki H, Kiyono M, Kougo S (1985) User-no Kaiwa-no Kata-wo Mochii-ta Shitsumon Outou System ('Selecting Topics in a Dialog according to Users Type'). Information Processing Society of Japan, WGNL 49-3 (in Japanese)

Takahashi M, Suzuki H, Kiyono M (1985) Production System wo Mochii-ta Nihongo Bunseisei ('Generation of Japanese Sentences by Production'), Proceedings of the 30th semiannual meeting of Information Processing Society of Japan: 1675-1676 (in Japanese)

Teramura H (1982) Nihongo-no Syntax to Imi I ('Syntax and Semantics of Japanese I'). Kuroshio, Tokyo (in Japanese)

Yasukawa H (1984) LFG System in Prolog. Proceedings of the 10th International Conference on Computational Linguistics: 358-361